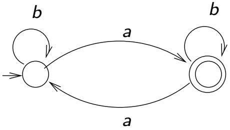
Overview of E0 222: Automata and Computability

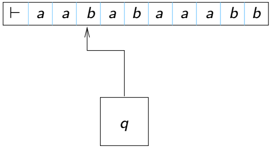
Deepak D'Souza

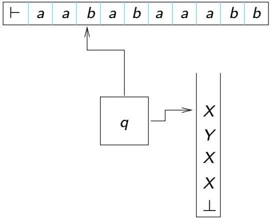
Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

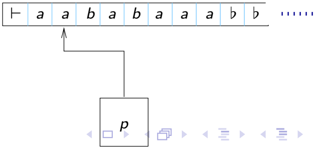
August 4, 2016

Different Kinds of "Automata" or "State Machines"



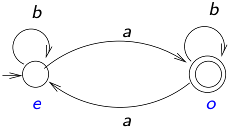


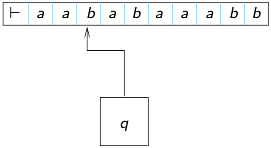


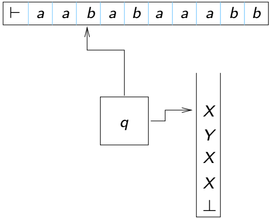


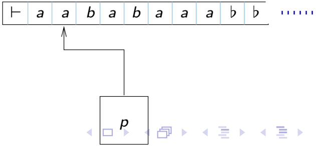
- Finite-State Automata
- Pushdown Automata
- Turing Machines

Different Kinds of “Automata” or “State Machines”









- Finite-State Automata
- Pushdown Automata
- Turing Machines

Kind of results we study in Automata Theory

- Expressive power of the models in terms of the class of languages they define.
  - Characterisations of this class of languages
    - Myhill-Nerode theorem.
    - Büchi's logical characterisation.
  - Necessary conditions these classes satisfy
    - Pumping Lemma and ultimate periodicity (for Regular/CFL).
    - Parikh's Theorem (for Context-Free Languages).
- Decision procedures
  - Emptiness problem
  - Language inclusion problem
  - Configuration reachability problem.
- Computability (most compelling notion of computable function is via Turing Machines), Rice's Theorem.

Why study automata theory?

Corner stone of many subjects in CS:

1. Compilers
   - Lexical analysis, parsing, regular expression search
2. Digital circuits (state minimization, analysis).
3. Complexity Theory (algorithmic hardness of problems)
4. Mathematical Logic
   - Decision procedures for logical problems.
5. Formal Verification
   - Configuration reachability
   - Is $L(\mathcal{A}) \subseteq L(\mathcal{B})$?

Uses in Verification

1. System models are natural extensions of automata models
   - Programs with no dynamic memory allocation, no procedures = Finite State systems.
   - No dynamic memory allocation = Pushdown systems.
   - General program = Turing machine.
   - Programs with integer variables = Counter machines.

   Decision procedures for emptiness, configuration reachability, etc, directly translate to decision procedures for programs.
2. To solve "model-checking" problem for logics that talk about infinite behaviour.

Uses in Logic

- Obtain decision procedure for satisfiability of a logic by translating a formula to an automaton and checking emptiness.
- Argue undecidability/incompleteness of a proof system.

What this course is about

What we study

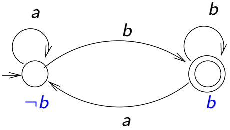
- Connections between Logic and Automata
  - Büchi's logical characterization of regular languages
  - Decision procedures for logic (Büchi, Presburger logic, Gödel's Incompleteness).
- Pushdown Systems
  - Parikh's theorem on semi-linearity of CFL's
  - Reachability in pushdown systems
  - Deterministic PDA's and complementation
  - Visibly Pushdown Automata
  - Decision procedures
- Automata on infinite words
- Automata on Trees

Büchi's logical characterisation of automata

- Describe properties of strings in a logical language
  Eg. "For all positions x in a word which are labelled a, there is a later position labelled b"

$$\forall x(Q_a(x) \Rightarrow \exists y(y > x \;\&\; Q_b(y))).$$

- DFA for the language:



- Büchi's result:

  A language is regular iff it is definable by a sentence in this logic.

First-Order logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \; \forall x \varphi, \; \neg, \; \& \,, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.

First-Order logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \ \&, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.

First-Order logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \ \&, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.

First-Order logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \; \forall x \varphi, \; \neg, \; \&, \vee.$$

- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.
  4. $\forall x \forall y ((x < y) \implies \exists z (x < z < y))$.

First-Order logic of $(\mathbb{N}, <)$.

- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \dots\}$.
- What you can say:

$$x < y, \quad \exists x \varphi, \ \forall x \varphi, \ \neg, \ \& \ , \vee.$$

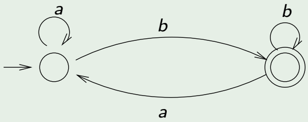
- Examples:
  1. $\forall x \exists y (x < y)$.
  2. $\forall x \exists y (y < x)$.
  3. $\exists x (\forall y (y \leq x))$.
  4. $\forall x \forall y ((x < y) \implies \exists z (x < z < y))$.
- Question: Is there an algorithm to decide if a given $\mathrm{FO}(\mathbb{N}, <)$ sentence is true or not?

  Büchi used automata to give such an algorithm.

Büchi automata

- Finite state automata that run over infinite words.
- How do we accept an *infinite* word? Acceptance mechanism proposed by Büchi: see if run visits a final state infinitely often.

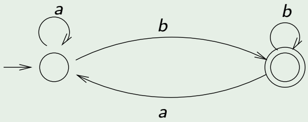
Büchi automaton for infinitely many b's



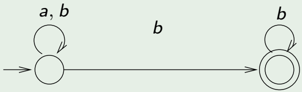
Büchi automata

- Finite state automata that run over infinite words.
- How do we accept an *infinite* word? Acceptance mechanism proposed by Büchi: see if run visits a final state infinitely often.

Büchi automaton for infinitely many *b*'s



Büchi automaton for finitely many *a*'s



Presburger Logic

- First-Order logic of $(\mathbb{N}, <, +)$.
- Interpreted over $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$.
- What you can say:

$$x + 2y < z + 1, \quad \exists x\varphi, \ \forall x\varphi, \ \neg, \ \&, \vee.$$

- Examples:
  1. $\forall x \forall y((x < y) \implies \exists z(x < z < y))$ (Also in FO(<)).
  2. Solutions to a system of linear inequalities: $\exists x \exists y(x + 2y \leq 1 \ \& \ x = y)$.
  3. "Every number is odd or even": $\forall x \exists y(x = 2y \vee x = 2y + 1)$.
- Studied by Mojzesz Presburger, who gave a sound and complete axiomatization, as well as a decision procedure for validity, circa 1929.

Overall idea

- Represent interpretation of variables as (rows of) binary strings

x	001111
y	100011
z	011100

- Construct automata over such words, that accept all satisfying assignments of the variables, for atomic formulas.
- Use closure properties of automata to inductively construct automata for more complex formulas.

Representing numbers as binary strings

- Represent the number 3 by "011" or "0011" or "00011" etc.
- The automata will read the strings from right to left.
- Will read a tuple of bits: For example for the formula $x \leq 2y + 1$ it will read inputs from the alphabet

$$\{0,1\}^2$$

which we represent as:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

- Thus, automaton constructed for a given formula will accept the reverse of actual interpretations.

Automaton for $x + 2y - 3z = 1$

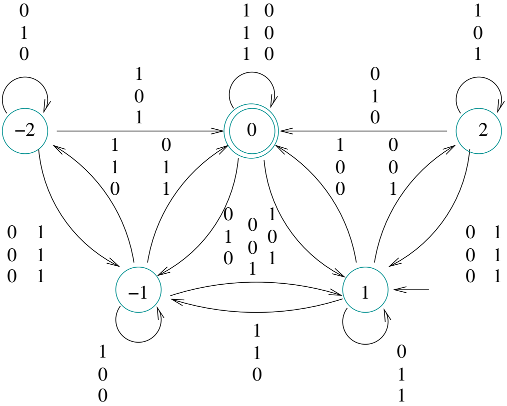
Accepting run on:

$x\,(=0):\quad 000$
$y\,(=2):\quad 010$
$z\,(=1):\quad 001$

$x\,(=15):\quad 001111$
$y\,(=35):\quad 100011$
$z\,(=28):\quad 011100$

but none on:

$x\,(=1):\quad 001$
$y\,(=2):\quad 010$
$z\,(=1):\quad 001$



Gödel's Incompleteness result

There cannot be a sound and complete proof system for first-order arithmetic.

What we can say in $\mathrm{FO}(\mathbb{N}, +, \cdot)$

- "Every number has a successor"

$$\forall n \exists m (m = n + 1).$$

- "Every number has a predecessor"

$$\forall n \exists m (n = m + 1).$$

- "There are only finitely many primes"

$$\exists n \forall p (prime(p) \implies p < n).$$

- "There are infinitely many primes"

$$\forall n \exists p (prime(p) \;\; \& \;\; p > n).$$

Gödel's Incompleteness result

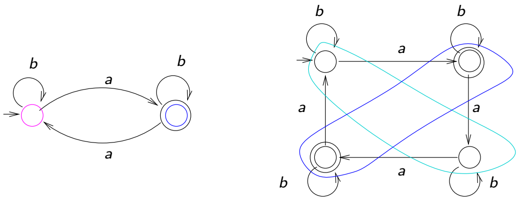
There cannot be a sound and complete proof system for first-order arithmetic.

Formal language-theoretic proof: $\mathrm{Th}(\mathbb{N}, +, .)$ is not even recursively enumerable.

Myhill-Nerode Theorem

Myhill-Nerode Theorem:

Every regular language has a canonical DFA accepting it.



Some consequences:

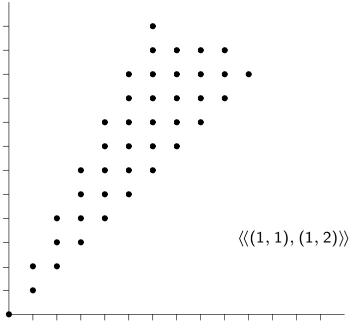
- Any DFA for L is a *refinement* of its canonical DFA.
- "minimal" DFA's for L are isomorphic.

Parikh's Theorem for CFL's

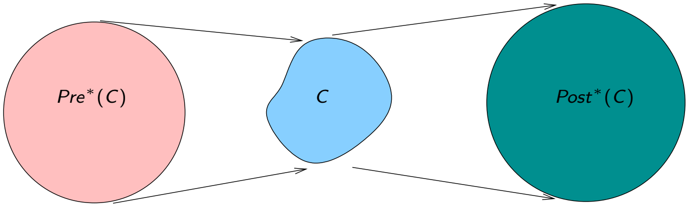
$\psi(w)$: "Letter-count" of a string w:

$$\text{Eg} : \psi(aabab) = (3, 2).$$

If L is a context-free language, then $\psi(L)$ is semi-linear (Every CFL is letter-equivalent to a regular language).



Reachable configurations of a Pushdown automaton



The set of reachable configurations of a Pushdown automaton is regular.

Useful for program analysis and verification of pushdown systems.

Course details

- Weightage: 40% assignments + seminar, 20% midsem exam, 40% final exam.
- Assignments to be done on your own.
- Dishonesty Policy: Any instance of copying in an assignment will fetch you a 0 in that assignment + one grade reduction.
- Seminar (in pairs) can be chosen from list on course webpage or your own topic.
- Course webpage: `www.csa.iisc.ernet.in/~deepakd/atc-2016/`
- Teaching assistants for the course: P. Ezudheen and Inzemamul Haque.
- Those interested in crediting/auditing please send me an email so that I can add you to the course mailing list.